Amendments to the Claims:

This listing of claims will replace all prior versions, and listings, of claims in the application:

Listing of Claims:

1. (currently amended) A method of maintaining http session data in a server system serving a network, said server system including at least one network server, said method comprising the steps of:

(1) storing in a database, session data for a plurality of sessions serviced by said at least one server;

(2) performing contemporaneous time out testing of particular session data for one of said plurality of sessions ~~of a particular session~~ every time a request is received for said particular session data prior to utilizing said particular session data, and ~~allowing~~ not invalidating said particular session data ~~to remain valid~~ in said database even ~~after~~ if said contemporaneous testing has indicated that the corresponding session has timed out; and

(3) performing an invalidation procedure on said session data in said database ~~for all of said sessions~~ at a particular time that is independent of when said contemporaneous testing is performed.

2. (original) The method of claim 1 wherein said session data comprises an HttpSession object of a Java servlet application program interface (API).

3. (original) The method of claim 2 wherein said Java servlet APIs are J2EE servlet APIs.

4. (original) The method of claim 2 wherein step (1) comprises the step of:

(1.1) creating an HttpSession object for a session upon initiation of said session.

5. (original) The method of claim 4 wherein step (1) further comprises the step of:

(1.2) updating said HttpSession object for said sessions as said session progresses.

6. (original) The method of claim 5 wherein said server system comprises a plurality of Java Virtual Machines (JVMs) of which different ones of said JVMs may service different http requests corresponding to a single http session and wherein said database is accessible to each of said JVMs.

7. (original) The method of claim 6 wherein step (1) further comprises the step of:

(1.3) storing said HttpSession object for each session handled by a JVM in a memory local to a server running said JVM;

(1.4) writing a copy of said HttpSession object for each session stored in said local memories to said database.

8. (original) The method of claim 7 wherein said plurality of JVMs run on a plurality of network servers.

9. (original) The method of claim 8 wherein said server system services the World Wide Web.

10. (original) The method of claim 1 wherein said particular time is a function of a periodic interval.

11. (original) The method of claim 10 wherein said periodic interval is a day and said particular time is a time of day.

12. (original) The method of claim 11 wherein said time of day is a time of day that a load on said database is expected to be low.

13. (original) The method of claim 1 further comprising the steps of:

(4) periodically determining a load on said database; and

wherein said particular time is a function of said determined load and a predetermined interval.

14. (original) The method of claim 1 wherein said invalidation procedure comprises invalidating all of said sessions stored in said database

15. (original) The method of claim 1 wherein said invalidation procedure comprises the steps of:

(3.1) for each session in said database, determining if said session has timed out;

(3.2) for each session that has timed out, invalidating the corresponding session data in said database.

16. (currently amended) A server system serving a network comprising:

at least one network server;

a memory;

a first computer program adapted to store in said memory session data for a plurality of sessions serviced by said at least one server;

a second computer program adapted to perform contemporaneous time out testing of particular session data for one of said plurality of sessions ~~of a particular session~~ every time a request is received for said particular session data prior to utilizing said particular session data, and further adapted to not invalidating ~~to allow~~ said particular session data ~~to remain valid~~ in said database even ~~after~~ if said contemporaneous testing has indicated that the corresponding session has timed out; and

a third computer program adapted to perform an invalidation procedure on said session data in said database ~~for all of said sessions~~ at a particular time that is independent of when said contemporaneous testing is performed.

17. (original) The system of claim 16 wherein said session data comprises an HttpSession object of a Java servlet application program interface (API).

18. (original) The system of claim 17 wherein said Java servlet APIs are J2EE servlet APIs.

19. (original) The system of claim 17 wherein said first program creates an HttpSession object for a session upon initiation of said session and updates said HttpSession object for said session as said session progress.

20. (previously presented) The system of claim 19 further comprising a plurality of Java Virtual Machines (JVMs) of which different ones of said JVMs may service different http requests corresponding to a single session and wherein said memory is accessible to each of said JVMs.

21. (original) The system of claim 20 wherein said first program stores said HttpSession object for each session handled by a JVM in a memory local to said JVM and writes a copy of said HttpSession object for each http session stored in said local memories to said database.

22. (original) The system of claim 21 wherein said at least one network server comprises a plurality of network servers and wherein different ones of said JVMs run on different ones of said network servers.

23. (original) The system of claim 22 wherein said server system services the World Wide Web.

24. (original) The system of claim 16 wherein said particular time is a function of a periodic interval.

25. (original) The system of claim 24 wherein said periodic interval is a day and said particular time is a time of day.

26. (original) The system of claim 25 wherein said time of day is a time of day that network traffic involving said server system is expected to be low.

27. (original) The system of claim 16 further comprising:

a computer program for determining a volume of network traffic involving said server system; and

wherein said particular time is a function of said network traffic involving said server system.

28. (original) The system of claim 27 wherein said particular time is further a function of a predetermined interval.

29. (previously presented) The system of claim 16 wherein said third program invalidates all of said sessions stored in said database at said particular time.

30. (previously presented) The system of claim 16 wherein, for each session in said database, said third program determines if said session has timed out and invalidates the session data corresponding to said sessions that have been determined to have timed out.

31. (currently amended) A method of maintaining HttpSession objects in a server system serving a network, said server system including a plurality of network servers running a plurality of Java Virtual Machines (JVMs), said method comprising the steps of:

(1) storing in a database accessible to all of said JVMs HttpSession objects for each session serviced by said JVMs;

(2) performing contemporaneous time out testing of a particular HttpSession object every time a request is received for said particular HttpSession object prior to utilizing said particular HttpSession object, and not invalidating ~~and allowing~~ said particular HttpSession object in said database ~~to remain valid~~ even ~~after~~ if said contemporaneous testing has indicated that the corresponding session has timed out; and

(3) performing an invalidation procedure on said HttpSession objects ~~for all of said http sessions~~ at a particular time that is independent of when said contemporaneous testing is performed.

32. (original) The method of claim 31 wherein said Java servlet APIs are J2EE servlet APIs.

33. (original) The method of claim 32 wherein step (1) comprises the steps of:

(1.1) creating an HttpSession object for a session upon initiation of said session and storing said HttpSession object in a memory local to a particular one of said JVMs upon initiation of said session;

(1.2) writing a copy of said HttpSession object for each session stored in said local memory to said database upon said creation;

(1.3) updating said HttpSession object for each said http session in said local memory as said session progresses.

(1.3) updating said copy of said corresponding HttpSession object in said database as said session progresses.

34. (original) The method of claim 32 wherein said particular time is a function of a periodic interval.

35. (original) The method of claim 34 wherein said periodic interval is a day and said particular time is a time of day when network traffic involving said server system is expected to be low.

36. (previously presented) The method of claim 31 further comprising the steps of:

(4) determining a volume of network traffic involving said server system; and

wherein said particular time is a function of said network traffic involving said server system.

37. (previously presented) The method of claim 31 wherein said invalidation procedure comprises invalidating all of said sessions stored in said database at said particular time.

38. (original) The method of claim 31 wherein said invalidation procedure comprises the steps of:

(3.1) for each HttpSession object in said database, determining if said corresponding session has timed out;

(3.2) invalidating each HttpSession object in said database that has timed out.